METHOD FOR DYNAMIC SELECTION FOR SECURE AND FIREWALL FRIENDLY COMMUNICATION PROTOCOLS BETWEEN MULTIPLE DISTRIBUTED MODULES

FIELD OF THE INVENTION

[0001] The present invention relates to communication protocols in a network, and more particularly to tunneling in a network using different communication protocols.

BACKGROUND OF THE INVENTION

[0002] Clients connected to a network often use a firewall for security purposes. The firewall controls incoming and outgoing communication in the network to protect resources of the network. In most applications, the firewall protects a private network from public access. Typically, the firewall is located at an entry point of the network and evaluates whether a particular communication user should be permitted access to the network.

[0003] The network may intend that certain outside users are able to access the network. These users may be authorized to access applications and data that are internal to the network. The network may implement a virtual private network (VPN) or a proxy server to provide access to the authorized users. If the network uses a proxy server to provide access, the proxy server must be tightly integrated with the network. The network cannot provide services without managing installation and configuration of the proxy server.

[0004] Additionally, the network may use tunneling to provide access to the authorized users. Tunneling refers to encapsulating information, such as a

data packet, in a different communication protocol. The firewall may be restricted to a certain type of protocol. Tunneling allows a data packet from an outside source to be encapsulated in a protocol that is accepted by the firewall.

[0005] VPN systems require interoperability between user hardware, software, and systems. Often, hardware must be from the same manufacturer or vendor in order to be interoperable. Additionally, the network may not intend that the outside user has access to all of the data in the network. However, bypassing the firewall may give the outside user access to the entire network. The VPN may not be able to limit the amount of data that an outside user can access.

SUMMARY OF THE INVENTION

[0006] A method for establishing communication in a network comprises determining communication data from a first network peer at a first tunnel. The communication data is registered with a lookup service. A communication request is received from a second network peer at the lookup service. The communication data of the first peer is provided to the second peer.

[0007] In another embodiment, a lookup service in a network comprises a first tunnel module that acquires communication data of a network peer. A registration table stores the communication data. A second tunnel module sends a communication request to the registration table, acquires the communication data from the registration table, and sends a communication attempt to the first tunnel based on the communication data.

[0008] Further areas of applicability of the present invention will become apparent from the detailed description provided hereinafter. It should be understood that the detailed description and specific examples, while indicating the preferred embodiment of the invention, are intended for purposes of illustration only and are not intended to limit the scope of the invention.

BRIEF DESCRIPTION OF THE DRAWINGS

[0009] The present invention will become more fully understood from the detailed description and the accompanying drawings, wherein:

[0010] Figure 1 is a functional block diagram of a protocol selection system according to the present invention;

[0011] Figure 2 is a functional block diagram of architecture of a tunnel system according to the present invention;

[0012] Figure 3 illustrates a lookup service according to the present invention;

[0013] Figure 4 illustrates the security authentication of a lookup service according to the present invention;

[0014] Figure 4A illustrates encrypted data transmission according to the present invention;

[0015] Figure 5 is a flow diagram of tunnel communication according to the present invention;

[0016] Figure 6 illustrates message retrieval according to the present invention;

[0017] Figure 7 illustrates TCP discovery according to the present invention; and

[0018] Figure 8 illustrates UDP discovery according to the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

[0019] The following description of the preferred embodiment(s) is merely exemplary in nature and is in no way intended to limit the invention, its application, or uses.

[0020] A protocol selection system 10 dynamically selects a secure communication protocol as shown in Figure 1. Tunnel systems 12, 14, 16, and 18 detect firewall restrictions of a network or client and determine appropriate tunnel mechanisms for communication. For example, the tunnel mechanism may be bidirectional. A bidirectional tunnel mechanism allows a client to communicate directly with another client. Another tunnel mechanism allows multiple clients to communicate through middleware. The middleware interacts with the tunnel systems and translates communication between the tunnel systems. For example, the middleware may be an intermediate server or other suitable application. In other words, the middleware acts as an intermediary between the tunnel systems, and therefore the clients. The tunnel systems 12, 14, 16, and 18 register communication information, including the firewall restrictions and the tunnel mechanisms, with a lookup service 20. The lookup service 20 may also determine which tunnel mechanism or which middleware to

use according to the tunnel systems involved with a particular message or communication request.

[0021] A tunnel system architecture 22 includes a tunnel module 24, a discovery module 26, and a registration module 28 as shown in Figure 2. The tunnel module 24 queries the discovery module 26 to determine communication information, including a communication address and tunnel communication capability of a client or network peer 30. The tunnel module 24 forwards the communication information to the registration module 28. The registration module 28 registers the communication information with the lookup service 20. The lookup service 20 includes a table of the communication information for each authorized peer in the system. When a peer makes a request to open communication, the lookup service 20 is responsible for providing the necessary communication addresses and protocol information to enable communication.

[0022] The lookup service 20 manages the communication information as shown in Figure 3. A tunnel must register its communication information before attempting to communicate with another tunnel. For example, peers 32 and 34 may be able to send and receive messages requests from behind a firewall. The peers 32 and 34 include tunnels 36, 38, 40, 42. The tunnels 36, 38, 40, 42 acquire logic names, such as a uniform resource identifier (URI), from a local resource. The local resource may be the discovery module 26 (as shown in Figure 2). The tunnels 36, 38, 40, 42 generate a global unique identifier from the URI. The tunnels 36, 38, 40, 42 register the logic names, unique identifiers, and communication addresses with the lookup service 20.

[0023] Conversely, peers 44 and 46 may only be able to send messages. The peers 44 and 46 include tunnels 48, 50. The tunnels 48, 50 acquire logic names and generate global unique identifiers. The tunnels 48, 50 register the logic names and unique identifiers with the lookup service 20.

[0024] Still referring to Figure 3, the lookup service 20 maintains a lookup table 52 that includes the registered information such as logic name 54, unique identifier 56, communication address (IP address) 58, port 60, and service capability link 62. The service capability links 62 store the links which are pointed to the location of a service capability descriptor. The service capability descriptor indicates the proxy capability. In other words, the service capability descriptor indicates whether the tunnel systems in peers 32 and 34 have the capability to act as a middleware server to maintain proxy queues for the tunnels 48 and 50. Additionally, the service capability descriptor may indicate proxy queue size, type of data that can be tunneled, security information, tunnel protocol type (TCP, UDP, RTP, etc), and address mapping information.

[0025] A tunnel with a communication address may accept incoming connection requests. Alternatively, a tunnel without a communication address does not have the capability to accept incoming connection requests. Often, a tunnel without a communication address is behind a firewall that blocks all incoming connection requests. Therefore, proxy queues are often used within the tunnels with communication addresses to compensate for the tunnels without communication addresses. For example, tunnels 48, 50, which are only able to send messages, do not have their own communication addresses. Any

messages sent to the tunnels 48, 50 are instead stored in proxy queues. The lookup service 20 searches the registration table 52 to determine which tunnel systems include proxy queue capabilities. The lookup service 20 assigns the communication address of the proxy queue-capable tunnels to the tunnels that do not have communication addresses. For example, the lookup service 20 may assign the communication address of the tunnel 36 to the tunnel 48. The tunnel 48 polls the proxy queue in the peer 32 to retrieve the messages intended for the tunnel 48. In this manner, proxy queues may be located locally or remotely with respect to each tunnel. In one embodiment, proxy queues may be located on a server with the lookup service. In another embodiment, proxy queues may be located on a dedicated proxy queue server.

[0026] Additionally, remote proxy queues may be created dynamically as requested by a peer. For example, before participating in communication, a peer determines whether it will need a remote proxy queue. The peer updates the lookup table 52 with information regarding the remote proxy queue, such as identity of the peer that requires the proxy queue.

[0027] The tunnels 36, 38, 40, 42 include a cache 64. Likewise, the tunnels 48, 50 include the cache 64. The cache 64 contains the data from the registration table 52. The tunnel module 24 (as shown in Figure 2) manages a queue of communication requests for the corresponding tunnel and the cache of the registration table. In the preferred embodiment, the lookup service 20 and the registration table 52 are implemented on a UDDI (universal description,

discovery, and integration) framework. However, other suitable implementations for the lookup service 20 and the registration table 52 may be used.

[0028] The lookup service 20 may authenticate the registration requests from tunnels 66, 68 as shown in Figure 4. The tunnels 66, 68 include a certificate 70. The tunnels 66, 68 include the certificate 70 with the registration information sent to the registration table 52. If the lookup service 20 verifies the certificate 70, the lookup service 20 returns a tunnel session identifier 72 to the tunnels 66, 68. The tunnel session identifier 72 may be associated with a time to indicate a lease time for the session. The tunnel module 24 maintains the validity of the tunnel session identifier 72 with the lookup service 20. The lookup service 20 authenticates communication requests from the tunnels 66, 68 based on the tunnel session identifier 72. For example, the tunnel 66 sends a communication request including the tunnel session identifier 72 to the tunnel 68. The tunnel 68 consults the lookup service 20 to verify the tunnel session identifier 72. If the tunnel session identifier 72 is valid, the tunnel 68 creates a proxy queue for the tunnel 66 and communication may begin.

[0029] In one embodiment, a first application 74 may intend to send encrypted data to a second application 76 as shown in Figure 4A. The application 74 must acquire a secured key from the second application 76 for symmetric encryption. Alternatively, the application 74 must acquire a public key for asymmetric encryption. However, if the application 76 is behind a firewall, the application 74 must use a tunnel module 78 to acquire the key. For example, an SSL (secure sockets layer) protocol proxy module 80 of the first application 74

may simulate an SSL proxy module 82 of the second application 76. The first application 74 sends out the public key 84 to the tunnel module 78 through the SSL proxy module 80. The tunnel module 78 sends the key 84 to the tunnel module 86 of the second application 76. The second application 76 can then acquire the key 84 through the SSL proxy module 82. Now, the first application 74 and the second application 76 may communicate through the tunnel modules 78 and 86 without using the SSL proxies.

[0030] Referring again to Figure 4, the lookup table 52 includes alternative communication methods and protocols that a particular tunnel or peer may use. In this manner, peers may select or alternate between communication protocols dynamically. For example, a peer may precede a UDP message with an HTTP header. The peer consults the lookup table 52 to determine the proper protocols for a target peer. The peer uses the information from the lookup table 52 to select the proper tunnel mechanism or protocol for communicating with the target peer. The peer may select a different tunnel mechanism or protocol for communicating with a different target peer.

[0031] A first tunnel initiates communication as shown in Figure 5. The first tunnel searches the registration table 52 to determine communication address and port information at step 100. The first tunnel creates a receive queue or proxy queue identified by a unique identifier at step 102. The tunnel module 24 acquires the data in the registration table 52 at step 104. The tunnel module 24 periodically updates the cache 64 with the data from the registration table 52. The tunnel module 24 updates the cache 64 by comparing the

registration data in the cache 64 with the registration table 52. If the data does not match, the tunnel module 24 refreshes the cache 64.

[0032] By consulting the registration data in the cache 64, the tunnel determines the logic name of a second tunnel at step 106. Using the logic name, the tunnel discovers the communication address of the second tunnel from the lookup service 20. The first tunnel sends a message to the second tunnel based on the second tunnel's communication address at step 108. The message includes the second tunnel's unique identifier. When the second tunnel receives the message, the second tunnel checks the unique identifier included with the message against the second tunnel's unique identifier at step 110. If the identifiers match, the second tunnel adds the message to a proxy or receive queue at step 112. If the identifiers do not match, the second tunnel rejects the message at step 114.

[0033] In another embodiment, a send only tunnel 120 may retrieve a message as shown in Figure 6. Receiving capability of the send only tunnel 120 is limited by a firewall. Tunnel 122 sends a message to the send only tunnel 120. Because the tunnel 120 is send only, the registration table 52 matches the tunnel 120 to the communication address of a send/receive tunnel 124. Therefore, the message from the tunnel 122, intended for the tunnel 120, is sent to the proxy queue of the tunnel 124. The tunnel 120 sends a request to the tunnel 124 to retrieve the message from the proxy queue. If the tunnels 120 and 122 are both send-only, the tunnels 120 and 122 may both retrieve messages from their corresponding proxy queues of the tunnel 124.

[0034] A discovery procedure for TCP (transmission control protocol) is shown in Figure 7. If a peer 128 is behind a NAT (network address translation) firewall, port mapping must be established. For example, the NAT 130 may assign a public port number to the peer 128 through UPnP (universal plug and play) protocol. Alternatively, the NAT 130 may assign a public port number to the peer 128 based on a configuration of a router. To establish port mapping through UPnP, the peer 128 sends a request to a UPnP router to map a public port to the service port of the peer 128. Upon receiving the request, the UPnP router allocates a public port to the peer 128.

[0035] The NAT 130 assigns the public port number when the peer 128 attempts discovery with the lookup service 20. The peer 128 also receives a URI (uniform resource identifier) from a local resource. The peer 128 registers the URI, the public port number, and a NAT public IP address with the lookup service 20. The lookup service 20 creates a table entry to store the URI, the public port number, and NAT IP address. The peer 128 registers the IP address and the public port. The lookup service 20 correlates the NAT IP address with the public port.

[0036] To initiate a communication session with the peer 128, another peer 129 looks up the URI of the peer 128 to determine the public port number and the NAT IP address. The peer 129 uses the public port number and the NAT IP address to initiate communication.

[0037] A discovery procedure for UDP (user datagram protocol) is shown in Figure 8. A tunnel 132 opens a UDP brokerage service port 134. The

UDP port 134 and the IP address of the tunnel 132 are mapped to a corresponding public port and IP address that are accessible to a public network 136. The corresponding public port and IP address are hereinafter referred to as the UBS (UDP brokerage service) port 138. The tunnel 132 may retrieve and maintain the mapping information using UPnP, manual configuration, or other suitable methods.

[0038] Still referring to Figure 8, the tunnel 132 registers the UBS port information with the lookup service 20. An external UDP application 140 interacts with the lookup service 20 to determine the UBS port information. The external application 140 opens its UDP port 142 for communication and sends a UDP connection setup request (UCSR) to the UBS port 138. The tunnel 132 compares and analyzes the origin IP address and port of the UCSR request and forwards the information to a destination application 144. The tunnel 132 may send a UDP connection setup reject message if the request cannot be granted. For example, the tunnel 132 may reject a request if resources are not available or if the external application 140 is not authorized to communicate.

[0039] The destination application 144 opens the UDP 134 port after receiving the UCSR origin IP address and port information. The destination application 144 responds through the UDP port 134 with a UDP connection setup confirm (UCSC) message. Alternatively, the destination application 144 may respond with a UDP connection setup rejects message if necessary. The external UDP application 140 analyzes the origin IP address and port information

from the UCSC message. In this manner, the external UDP application 140 may point future communication directly to the destination application 144.

[0040] The external application 140 sends a UDP connection ready (UCR) message to the destination application 144. The destination application 144 responds with a UCR message. The external application 140 and the destination application 144 may now begin communication. The external application 140 and/or the destination application 144 may maintain a timer. If no accepted messages are received during a period defined by the timer, the applications may discontinue communication.

[0041] The description of the invention is merely exemplary in nature and, thus, variations that do not depart from the gist of the invention are intended to be within the scope of the invention. Such variations are not to be regarded as a departure from the spirit and scope of the invention.